Gowin GW1N-2 Hardened MIPI D-PHY RX

用户指南

IPUG778-1.01,2021-07-07

免责声明

本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和／或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改文档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

版本信息

日期	版本	说明
2021/06/01	1.0	初始版本。
2021/07/07	1.01	新增附录 A MIPI D-PHY 速率表。

目录

图目录

表目录

1 关于本手册

1.1 手册内容

Gowin MIPI D-PHY RX 用户指南主要内容包括功能特点、端口描述、配置调用等。主要用于帮助用户快速了解 Gowin MIPI D-PHY RX/TX 的产品特性、特点及使用方法。

1.2 相关文档

通过登录高云半导体网站 www.gowinsemi.com.cn 可以下载、查看以下相关文档：

1. DS100，GW1N 系列 FPGA 产品数据手册
2. UG103, GW1N 系列 FPGA 产品封装与管脚手册
3. UG171, GW1N-2 器件 Pinout 手册

1.3 术语、缩略语

表 1-1 中列出了本手册中出现的相关术语、缩略语及相关释义。

表 1-1 术语、缩略语

术语、缩略语	全称	含义
MIPI	Mobile Industry Processor Interface	移动行业处理器接口
HS	High Speed	高速
LP	Low Power	低功耗
I/O	Input/Output	输入/输出
DSI	Display Serial Interface	串行显示接口
CSI	Camera Serial Interface	串行摄像头接口

1.4 技术支持与反馈

高云半导体提供全方位技术支持，在使用过程中如有任何疑问或建议，可直接与公司联系：

网址：www.gowinsemi.com.cn

E-mail：support@gowinsemi.com

Tel：+86 755 8262 0391

2 概述

GW1N-2 器件包含硬核 MIPI D-PHY RX，支持标准《MIPI Alliance Standard for D-PHY Specification》，版本 2.1。该 D-PHY 适用于串行显示接口（Display Serial Interface，DSI）和和串行摄像头接口（Camera Serial Interface，CSI-2）。

2.1 特性

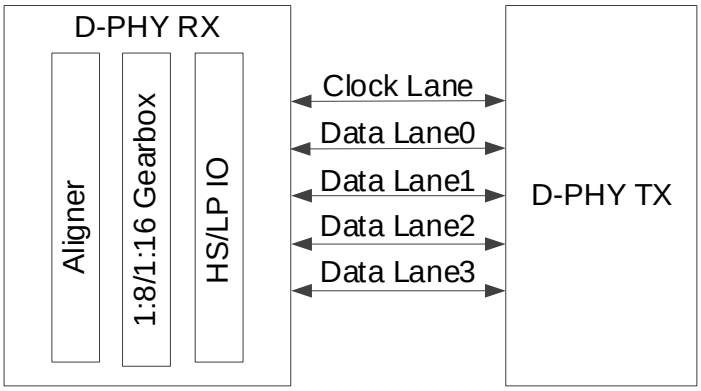
MIPI D-PHY RX 主要特性如下：

- 支持单向高速(HS，High-speed)模式，传输速率最高可达 8 Gbps (四个数据通道)。
- 支持最多四个数据通道和一个时钟通道。
- 支持双向低功耗(LP，Low-power)操作模式，数据传输速率为 10Mbps。
- 支持高速同步、位和通道对齐
- 支持 MIPI D-PHY RX 1:8 模式与 1:16 模式。
- 支持 MIPI DSI 和 MIPI CSI-2 链路层。
- IO Bank6 支持 MIPI D-PHY RX。

2.2 功能描述

MIPI D-PHY RX 主要包含 HS/LP IO, 1:8/1:16 Gearbox 及 Aligner 三部分，结构示意图如图 2-1 所示。

图 2-1 MIPI D-PHY RX 结构示意图



HS/LP I/O

支持 ODT 动态切换，支持 LP TX/RX, HS RX 动态切换。

1:8/1:16 Gearbox

支持 8 bits 或者 16 bits 位宽模式，可以通过 HS_8BIT_MODE 配置

Note!

MIPI_DPHY_RX 端口介绍请参考 3 MIPI D-PHY RX 原语 > 端口介绍。

Aligner

支持 Word align 和 Lane align，都通过 MIPI_DPHY_RX 端口 WALIGN_BY 和 LALIGN_EN 配置是否使能。其中 Word align 的 key 可以用户自定义（ALIGN_BYTE），且支持 2 bytes 和 3 bytes 模式，通过 MIPI_DPHY_RX 端口 ONE_BYTE0_MATCH 配置。

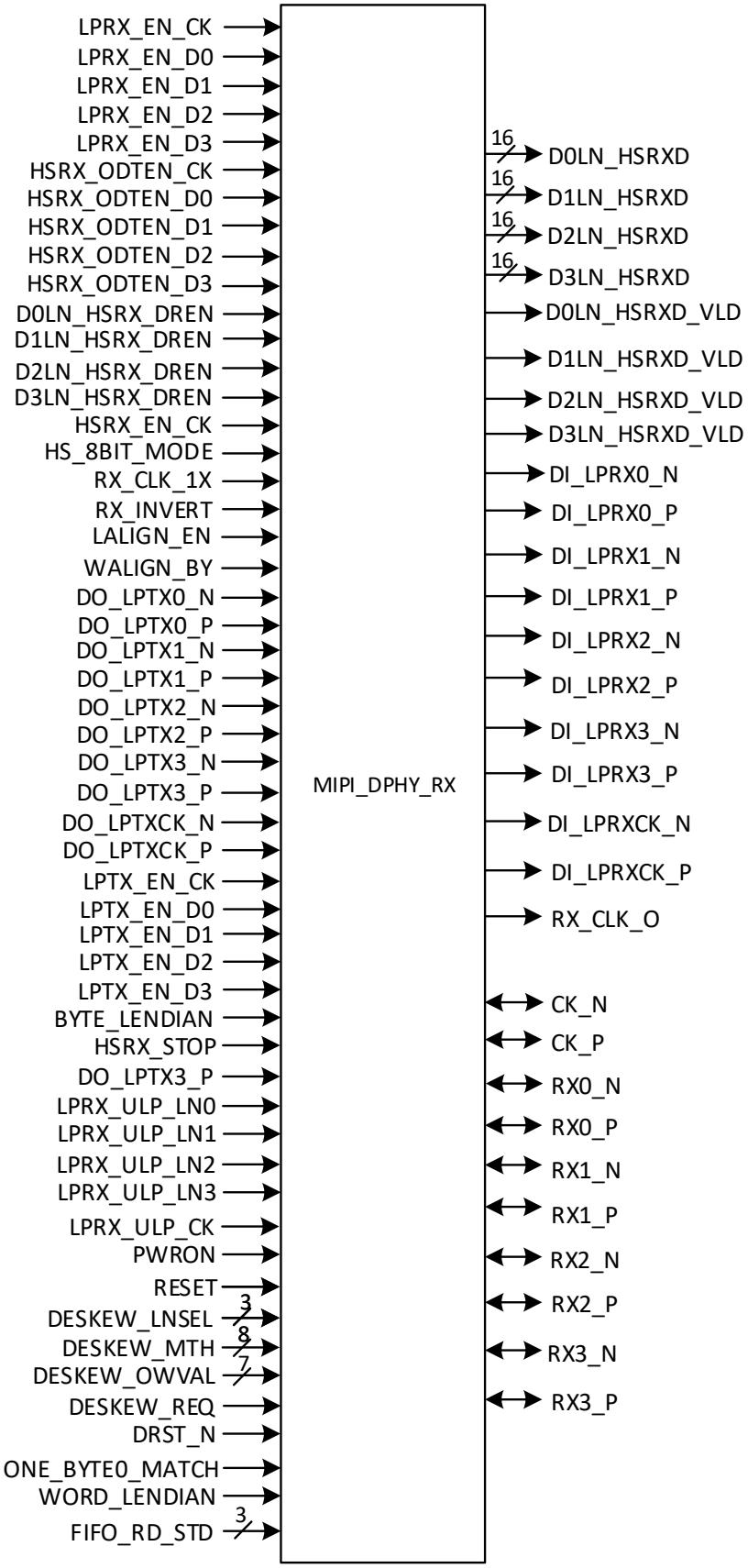
Note!

MIPI_DPHY_RX 端口介绍请参考 3 MIPI D-PHY RX 原语 > 端口介绍。

3 MIPI D-PHY RX 原语

3.1 端口示意图

图 3-1 MIPI_DPHY_RX 端口示意图



3.2 端口介绍

表 3-1 MIPI_DPHY_RX 端口介绍

端口	I/O	描述
LPRX_EN_CK	input	CK Lane: 1'b1 LPRX enabled
LPRX_EN_D0	input	Data Lane0: ● 1'b1 LPRX enabled ● 1'b0 LPRX disabled
LPRX_EN_D1	input	Data Lane1: ● 1'b1 LPRX enabled ● 1'b0 LPRX disabled
LPRX_EN_D2	input	Data Lane2: ● 1'b1 LPRX enabled ● 1'b0 LPRX disabled
LPRX_EN_D3	input	Data Lane3: ● 1'b1 LPRX enabled ● 1'b0 LPRX disabled
HSRX_ODTEN_CK	input	CK Lane HSRX ODT enabled: ● 1'b1 LPRX enabled ● 1'b0 LPRX disabled
HSRX_ODTEN_D0	input	Data Lane0 HSRX ODT enabled: ● 1'b1 ODT enabled ● 1'b0 ODT disabled
HSRX_ODTEN_D1	input	Data Lane1 HSRX ODT enabled: ● 1'b1 ODT enabled ● 1'b0 ODT disabled
HSRX_ODTEN_D2	input	Data Lane2 HSRX ODT enabled: ● 1'b1 ODT enabled ● 1'b0 ODT disabled
HSRX_ODTEN_D3	input	Data Lane3 HSRX ODT enabled: ● 1'b1 ODT enabled ● 1'b0 ODT disabled
D0LN_HSRX_DREN	input	Data Lane0 HSRX driver enabled: ● 1'b1 driver enabled ● 1'b0 driver disabled
D1LN_HSRX_DREN	input	Data Lane1 HSRX driver enabled： ● 1'b1 driver enabled ● 1'b0 driver disabled
D2LN_HSRX_DREN	input	Data Lane2 HSRX driver enabled： ● 1'b1 driver enabled ● 1'b0 driver disabled
D3LN_HSRX_DREN	input	Data Lane3 HSRX driver enabled ● 1'b1 driver enabled

端口	I/O	描述
		● 1'b0 driver disabled
HSRX_EN_CK	input	CK Lane: 1'b1 HSRX enabled
HS_8BIT_MODE	input	Selection of data width to Fabric ● 1'b1:8bit output mode ● 1'b0:16bit output mode
RX_CLK_1X	input	1X clock from fabric, max 93.75MHz@1.5Gbps
RX_INVERT	input	data polarity selection
LALIGN_EN	input	lane aligner enable ● 1'b1: lane aligner is enabled ● 1'b0: lane aligner is disabled
WALIGN_BY	input	word aligner bypass ● 1'b1: bypass ● 1'b0: word aligner is enabled
DO_LPTX0_N	input	Data Lane0 Complement Pad LPTX output
DO_LPTX0_P	input	Data Lane0 True Pad LPTX output
DO_LPTX1_N	input	Data Lane1 Complement Pad LPTX output
DO_LPTX1_P	input	Data Lane1 True Pad LPTX output
DO_LPTX2_N	input	Data Lane2 Complement Pad LPTX output
DO_LPTX2_P	input	Data Lane2 True Pad LPTX output
DO_LPTX3_N	input	Data Lane3 Complement Pad LPTX output
DO_LPTX3_P	input	Data Lane3 True Pad LPTX output
DO_LPTXCK_N	input	CK Lane Complement Pad LPTX output
DO_LPTXCK_P	input	CK Lane True Pad LPTX output
LPTX_EN_CK	input	CK Lane: 1'b1 LPTX enabled
LPTX_EN_D0	input	Data Lane0: 1'b1 LPTX enabled
LPTX_EN_D1	input	Data Lane1: 1'b1 LPTX enabled
LPTX_EN_D2	input	Data Lane2: 1'b1 LPTX enabled
LPTX_EN_D3	input	Data Lane3: 1'b1 LPTX enabled
BYTE_LENDIAN	input	bit data Littlendian/Bigendian of 8bit ● 1'b1: Littlendian, the bit entered first is in the low position of byte ● 1'b0: Bigendian, the bit entered first is in the high position of byte
HSRX_STOP	input	HSRX Clock Stop Signal for synchronization
LPRX_ULP_LN0	input	Data Lane0 LPRX ULP Control: 1'b1 enable
LPRX_ULP_LN1	input	Data Lane1 LPRX ULP Control: 1'b1 enable
LPRX_ULP_LN2	input	Data Lane2 LPRX ULP Control: 1'b1 enable
LPRX_ULP_LN3	input	Data Lane3 LPRX ULP Control: 1'b1 enable
LPRX_ULP_CK	input	CK Lane LPRX ULP Control: 1'b1 enable
PWRON	input	PowerOn Control: ● 1'b1: HSRX on ● 1'b0: HSRX off to standby in low power state

端口	I/O	描述
RESET	input	Reset signal: 1'b1: reset all;
DESKEW_LNSEL[2:0]	input	selection of lane to config delay overwrite value and make lane in deskew delay overwrite mode. 3'h0: NULL；select deskew error report output 3'h1: data lane0 3'h2: data lane1 3'h3: data lane2 3'h4: data lane3 3'h5: clock lane 3'h6: select lane aligner fifo write full or read empty error report output 3'h7: select 4 data lane fifo write full error report output **NOTE**! If any lane's deskew delay overwrite mode is ever active, a reset is a must before using this lane's deskew function normally.
DESKEW_MTH [7:0]	input	counter threathold for searching one edge
DESKEW_OWVAL [6:0]	input	IN deskew delay overwrite mode : set delay overwrite value of lane: 0~123. **NOTE**! This value is only registered into module when the lane is selected to config delay overwrite value. IN normal mode: [6]: deskew_rst_bypass mode 0: deskew's rstn is the same as drst_n 1: deskew's rstn is not controlled by drst_n [5]: deskew_fast_mode 0: deskew is in initial mode 1: deskew is in fast mode [4]: deskew_en_low_delay 0: low delay is not enabled 1: low delay is enabled [3:0]: deskew_fast_loop_time
DESKEW_REQ	input	deskew function request to all data lanes
DRST_N	input	digital reset, active low
ONE_BYTE0_MATCH	input	byte count match in word aligner 1'b1: three bytes, one more byte of 0 match before KEY, word aligner is searching {KEY,8'b0,8'b0}~{KEY[7],{KEY[6:0],1'b0},8'b0} (Littlendian) 1'b0: two bytes, at most one byte of 0 match before KEY, word aligner is searching {KEY,8'b0}~{KEY[7],{KEY[6:0],1'b0}}(Littlendian)
WORD_LENDIAN	input	data little/big endian of dual word(16bit, 8bit/word). Not used in 8bit data output mode.

端口	I/O	描述
		1'b1: Littlendian, first word in low byte of 16bit. Lane aligner is searching:{8'hx,KEY}; 1'b0: Bigendian, first word in high byte of 16bit. Lane aligner is searching:{KEY,8'hx};
FIFO_RD_STD [2:0]	input	FIFO read threadhold; Can only be 1 in 8 bits mode.
D0LN_HSRXD[15:0]	output	data lane0 HS data output to fabric
D1LN_HSRXD[15:0]	output	data lane1 HS data output to fabric
D2LN_HSRXD[15:0]	output	data lane2 HS data output to fabric
D3LN_HSRXD[15:0]	output	data lane3 HS data output to fabric
D0LN_HSRXD_VLD	output	data lane0 HS data output valid to fabric
D1LN_HSRXD_VLD	output	data lane1 HS data output valid to fabric
D2LN_HSRXD_VLD	output	data lane2 HS data output valid to fabric
D3LN_HSRXD_VLD	output	data lane3 HS data output valid to fabric
DI_LPRX0_N	output	Data Lane0 Complement Pad LPRX input
DI_LPRX0_P	output	Data Lane0 True Pad LPRX input
DI_LPRX1_N	output	Data Lane1 Complement Pad LPRX input
DI_LPRX1_P	output	Data Lane1 True Pad LPRX input
DI_LPRX2_N	output	Data Lane2 Complement Pad LPRX input
DI_LPRX2_P	output	Data Lane2 True Pad LPRX input
DI_LPRX3_N	output	Data Lane3 Complement Pad LPRX input
DI_LPRX3_P	output	Data Lane3 True Pad LPRX input
DI_LPRXCK_N	output	CK Lane Complement Pad LPRX input
DI_LPRXCK_P	output	CK Lane True Pad LPRX input
RX_CLK_O	output	RX output 1X clock, max 93.75MHz@1.5Gbps
CK_N	inout	CK Lane Complement Input
CK_P	inout	CK Lane True Input
RX0_N	inout	Data Lane 0 Complement Input
RX0_P	inout	Data Lane 0 True Input
RX1_N	inout	Data Lane 1 Complement Input
RX1_P	inout	Data Lane 1 True Input
RX2_N	inout	Data Lane 2 Complement Input
RX2_P	inout	Data Lane 2 True Input
RX3_N	inout	Data Lane 3 Complement Input
RX3_P	inout	Data Lane 3 True Input

3.3 参数介绍

表 3-2 MIPI_DPHY_RX 参数介绍

参数	默认	描述
ALIGN_BYTE	8'b10111000	KEY for word aligner and lane aligner
MIPI_LANE0_EN	1'b0	1'b1, Lane0 on;
MIPI_LANE1_EN	1'b0	1'b1, Lane1 on;
MIPI_LANE2_EN	1'b0	1'b1, Lane2 on;
MIPI_LANE3_EN	1'b0	1'b1, Lane3 on;
MIPI_CK_EN	1'b1	CK Lane Select: 1'b1 CK Lane on
SYNC_CLK_SEL	1'b0	Select clock source for HS lane output data: ● 0: select fabric input clock rx_clk_1x ● 1: select output clock RX_CLK_O

3.4 原语例化

Verilog 例化：

```
MIPI_DPHY_RX mipi_dphy_rx_inst (
    .D0LN_HSRXD(d0ln_hsrxd),
    .D1LN_HSRXD(d1ln_hsrxd),
    .D2LN_HSRXD(d2ln_hsrxd),
    .D3LN_HSRXD(d3ln_hsrxd),
    .D0LN_HSRXD_VLD(d0ln_hsrxd_vld),
    .D1LN_HSRXD_VLD(d1ln_hsrxd_vld),
    .D2LN_HSRXD_VLD(d2ln_hsrxd_vld),
    .D3LN_HSRXD_VLD(d3ln_hsrxd_vld),
    .DI_LPRX0_N(di_lprx0_n),
    .DI_LPRX0_P(di_lprx0_p),
    .DI_LPRX1_N(di_lprx1_n),
    .DI_LPRX1_P(di_lprx1_p),
    .DI_LPRX2_N(di_lprx2_n),
    .DI_LPRX2_P(di_lprx2_p),
    .DI_LPRX3_N(di_lprx3_n),
    .DI_LPRX3_P(di_lprx3_p),
    .DI_LPRXCK_N(di_lprxck_n),
    .DI_LPRXCK_P(di_lprxck_p),
    .RX_CLK_O(rx_clk_o),
    .CK_N(ck_n),
    .CK_P(ck_p),
    .RX0_N(rx0_n),
    .RX0_P(rx0_p),
    .RX1_N(rx1_n),
    .RX1_P(rx1_p),
```

```
.RX2_N(rx2_n),
.RX2_P(rx2_p),
.RX3_N(rx3_n),
.RX3_P(rx3_p),
.LPRX_EN_CK(lprx_en_ck),
.LPRX_EN_D0(lprx_en_d0),
.LPRX_EN_D1(lprx_en_d1),
.LPRX_EN_D2(lprx_en_d2),
.LPRX_EN_D3(lprx_en_d3),
.HSRX_ODTEN_CK(hsrx_odten_ck),
.HSRX_ODTEN_D0(hsrx_odten_d0),
.HSRX_ODTEN_D1(hsrx_odten_d1),
.HSRX_ODTEN_D2(hsrx_odten_d2),
.HSRX_ODTEN_D3(hsrx_odten_d3),
.D0LN_HSRX_DREN(d0ln_hsrx_dren),
.D1LN_HSRX_DREN(d1ln_hsrx_dren),
.D2LN_HSRX_DREN(d2ln_hsrx_dren),
.D3LN_HSRX_DREN(d3ln_hsrx_dren),
.HSRX_EN_CK(hsrx_en_ck),
.HS_8BIT_MODE(hs_8bit_mode),
.RX_CLK_1X(rx_clk_1x),
.RX_INVERT(rx_invert),
.LALIGN_EN(lalign_en),
.WALIGN_BY(walign_by),
.DO_LPTX0_N(do_lptx0_n),
.DO_LPTX0_P(do_lptx0_p),
.DO_LPTX1_N(do_lptx1_n),
.DO_LPTX1_P(do_lptx1_p),
.DO_LPTX2_N(do_lptx2_n),
.DO_LPTX2_P(do_lptx2_p),
.DO_LPTX3_N(do_lptx3_n),
.DO_LPTX3_P(do_lptx3_p),
.DO_LPTXCK_N(do_lptxck_n),
.DO_LPTXCK_P(do_lptxck_p),
.LPTX_EN_CK(lptx_en_ck),
.LPTX_EN_D0(lptx_en_d0),
.LPTX_EN_D1(lptx_en_d1),
.LPTX_EN_D2(lptx_en_d2),
.LPTX_EN_D3(lptx_en_d3),
.BYTE_LENDIAN(byte_lendian),
.HSRX_STOP(hsrx_stop),
.LPRX_ULP_LN0(lprx_ulp_ln0),
.LPRX_ULP_LN1(lprx_ulp_ln1),
.LPRX_ULP_LN2(lprx_ulp_ln2),
```

```
        .LPRX_ULP_LN3(lprx_ulp_ln3),
        .LPRX_ULP_CK(lprx_ulp_ck),
        .PWRON(pwron),
        .RESET(reset),
        .DESKEW_LNSEL(deskew_lnsel),
        .DESKEW_MTH(deskew_mth),
        .DESKEW_OWVAL(deskew_owval),
        .DESKEW_REQ(deskew_req),
        .DRST_N(drst_n),
        .ONE_BYTE0_MATCH(one_byte0_match),
        .WORD_LENDIAN(word_lendian),
        .FIFO_RD_STD(fifo_rd_std)
    );

    defparam mipi_dphy_rx_inst.MIPI_LANE0_EN = 1'b1;
    defparam mipi_dphy_rx_inst.MIPI_LANE1_EN = 1'b1;
    defparam mipi_dphy_rx_inst.MIPI_LANE2_EN = 1'b1;
    defparam mipi_dphy_rx_inst.MIPI_LANE3_EN = 1'b1;
    defparam mipi_dphy_rx_inst.MIPI_CK_EN = 1'b1;
    defparam mipi_dphy_rx_inst.SYNC_CLK_SEL = 1'b1;
    defparam mipi_dphy_rx_inst.ALIGN_BYTE = 8'b10111000;
```

Vhdl 例化：

```
    COMPONENT MIPI_DPHY_RX
            GENERIC (
                MIPI_LANE0_EN: in bit := '0';
                MIPI_LANE1_EN: in bit := '0';
                MIPI_LANE2_EN: in bit := '0';
                MIPI_LANE3_EN: in bit := '0';
                MIPI_CK_EN: in bit := '1';
                SYNC_CLK_SEL: in bit := '1';
                ALIGN_BYTE: in bit_vector := "10111000"
            );
            PORT(
                D0LN_HSRXD: out std_logic_vector(15 downto 0);
                D1LN_HSRXD: out std_logic_vector(15 downto 0);
                D2LN_HSRXD: out std_logic_vector(15 downto 0);
                D3LN_HSRXD: out std_logic_vector(15 downto 0);
                D0LN_HSRXD_VLD: out std_logic;
                D1LN_HSRXD_VLD: out std_logic;
                D2LN_HSRXD_VLD: out std_logic;
                D3LN_HSRXD_VLD: out std_logic;
                DI_LPRX0_N: out std_logic;
                DI_LPRX0_P: out std_logic;
```

```
DI_LPRX1_N: out std_logic;
DI_LPRX1_P: out std_logic;
DI_LPRX2_N: out std_logic;
DI_LPRX2_P: out std_logic;
DI_LPRX3_N: out std_logic;
DI_LPRX3_P: out std_logic;
DI_LPRXCK_N: out std_logic;
DI_LPRXCK_P: out std_logic;
RX_CLK_O: out std_logic;
CK_N: inout std_logic;
CK_P: inout std_logic;
RX0_N: inout std_logic;
RX0_P: inout std_logic;
RX1_N: inout std_logic;
RX1_P: inout std_logic;
RX2_N: inout std_logic;
RX2_P: inout std_logic;
RX3_N: inout std_logic;
RX3_P: inout std_logic;
LPRX_EN_CK: in std_logic;
LPRX_EN_D0: in std_logic;
LPRX_EN_D1: in std_logic;
LPRX_EN_D2: in std_logic;
LPRX_EN_D3: in std_logic;
HSRX_ODTEN_CK: in std_logic;
HSRX_ODTEN_D0: in std_logic;
HSRX_ODTEN_D1: in std_logic;
HSRX_ODTEN_D2: in std_logic;
HSRX_ODTEN_D3: in std_logic;
D0LN_HSRX_DREN: in std_logic;
D1LN_HSRX_DREN: in std_logic;
D2LN_HSRX_DREN: in std_logic;
D3LN_HSRX_DREN: in std_logic;
HSRX_EN_CK: in std_logic;
HS_8BIT_MODE: in std_logic;
RX_CLK_1X: in std_logic;
RX_INVERT: in std_logic;
LALIGN_EN: in std_logic;
WALIGN_BY: in std_logic;
DO_LPTX0_N: in std_logic;
DO_LPTX0_P: in std_logic;
DO_LPTX1_N: in std_logic;
DO_LPTX1_P: in std_logic;
DO_LPTX2_N: in std_logic;
```

```
                DO_LPTX2_P: in std_logic;
                DO_LPTX3_N: in std_logic;
                DO_LPTX3_P: in std_logic;
                DO_LPTXCK_N: in std_logic;
                DO_LPTXCK_P: in std_logic;
                LPTX_EN_CK: in std_logic;
                LPTX_EN_D0: in std_logic;
                LPTX_EN_D1: in std_logic;
                LPTX_EN_D2: in std_logic;
                LPTX_EN_D3: in std_logic;
                BYTE_LENDIAN: in std_logic;
                HSRX_STOP: in std_logic;
                LPRX_ULP_LN0: in std_logic;
                LPRX_ULP_LN1: in std_logic;
                LPRX_ULP_LN2: in std_logic;
                LPRX_ULP_LN3: in std_logic;
                LPRX_ULP_CK: in std_logic;
                PWRON: in std_logic;
                RESET: in std_logic;
                DESKEW_LNSEL: in std_logic_vector(2 downto 0);
                DESKEW_MTH: in std_logic_vector(7 downto 0);
                DESKEW_OWVAL: in std_logic_vector(6 downto 0);
                DESKEW_REQ: in std_logic;
                DRST_N: in std_logic;
                ONE_BYTE0_MATCH: in std_logic;
                WORD_LENDIAN: in std_logic;
                FIFO_RD_STD: in std_logic_vector(2 downto 0)
        );
    END COMPONENT;
        mipi_dphy_rx_inst: MIPI_DPHY_RX
            GENERIC MAP(
                MIPI_LANE0_EN => '1',
                MIPI_LANE1_EN => '1',
                MIPI_LANE2_EN => '1',
                MIPI_LANE3_EN => '1',
                MIPI_CK_EN => '1',
                SYNC_CLK_SEL => '1',
                ALIGN_BYTE => "10111000"
            )
            PORT MAP (
                D0LN_HSRXD => d0ln_hsrxd,
                D1LN_HSRXD => d1ln_hsrxd,
                D2LN_HSRXD => d2ln_hsrxd,
                D3LN_HSRXD => d3ln_hsrxd,
```

```
D0LN_HSRXD_VLD => d0ln_hsrxd_vld,
D1LN_HSRXD_VLD => d1ln_hsrxd_vld,
D2LN_HSRXD_VLD => d2ln_hsrxd_vld,
D3LN_HSRXD_VLD => d3ln_hsrxd_vld,
DI_LPRX0_N => di_lprx0_n,
DI_LPRX0_P => di_lprx0_p,
DI_LPRX1_N => di_lprx1_n,
DI_LPRX1_P => di_lprx1_p,
DI_LPRX2_N => di_lprx2_n,
DI_LPRX2_P => di_lprx2_p,
DI_LPRX3_N => di_lprx3_n,
DI_LPRX3_P => di_lprx3_p,
DI_LPRXCK_N => di_lprxck_n,
DI_LPRXCK_P => di_lprxck_p,
RX_CLK_O => rx_clk_o,
CK_N => ck_n,
CK_P => ck_p,
RX0_N => rx0_n,
RX0_P => rx0_p,
RX1_N => rx1_n,
RX1_P => rx1_p,
RX2_N => rx2_n,
RX2_P => rx2_p,
RX3_N => rx3_n,
RX3_P => rx3_p,
LPRX_EN_CK => lprx_en_ck,
LPRX_EN_D0 => lprx_en_d0,
LPRX_EN_D1 => lprx_en_d1,
LPRX_EN_D2 => lprx_en_d2,
LPRX_EN_D3 => lprx_en_d3,
HSRX_ODTEN_CK => hsrx_odten_ck,
HSRX_ODTEN_D0 => hsrx_odten_d0,
HSRX_ODTEN_D1 => hsrx_odten_d1,
HSRX_ODTEN_D2 => hsrx_odten_d2,
HSRX_ODTEN_D3 => hsrx_odten_d3,
D0LN_HSRX_DREN => d0ln_hsrx_dren,
D1LN_HSRX_DREN => d1ln_hsrx_dren,
D2LN_HSRX_DREN => d2ln_hsrx_dren,
D3LN_HSRX_DREN => d3ln_hsrx_dren,
HSRX_EN_CK => hsrx_en_ck,
HS_8BIT_MODE => hs_8bit_mode,
RX_CLK_1X => rx_clk_1x,
RX_INVERT => rx_invert,
LALIGN_EN => lalign_en,
```

```
        WALIGN_BY => walign_by,
        DO_LPTX0_N => do_lptx0_n,
        DO_LPTX0_P => do_lptx0_p,
        DO_LPTX1_N => do_lptx1_n,
        DO_LPTX1_P => do_lptx1_p,
        DO_LPTX2_N => do_lptx2_n,
        DO_LPTX2_P => do_lptx2_p,
        DO_LPTX3_N => do_lptx3_n,
        DO_LPTX3_P => do_lptx3_p,
        DO_LPTXCK_N => do_lptxck_n,
        DO_LPTXCK_P => do_lptxck_p,
        LPTX_EN_CK => lptx_en_ck,
        LPTX_EN_D0 => lptx_en_d0,
        LPTX_EN_D1 => lptx_en_d1,
        LPTX_EN_D2 => lptx_en_d2,
        LPTX_EN_D3 => lptx_en_d3,
        BYTE_LENDIAN => byte_lendian,
        HSRX_STOP => hsrx_stop,
        LPRX_ULP_LN0 => lprx_ulp_ln0,
        LPRX_ULP_LN1 => lprx_ulp_ln1,
        LPRX_ULP_LN2 => lprx_ulp_ln2,
        LPRX_ULP_LN3 => lprx_ulp_ln3,
        LPRX_ULP_CK => lprx_ulp_ck,
        PWRON => pwron,
        RESET => reset,
        DESKEW_LNSEL => deskew_lnsel,
        DESKEW_MTH => deskew_mth,
        DESKEW_OWVAL => deskew_owval,
        DESKEW_REQ => deskew_req,
        DRST_N => drst_n,
        ONE_BYTE0_MATCH => one_byte0_match,
        WORD_LENDIAN => word_lendian,
        FIFO_RD_STD => fifo_rd_std
    );
```

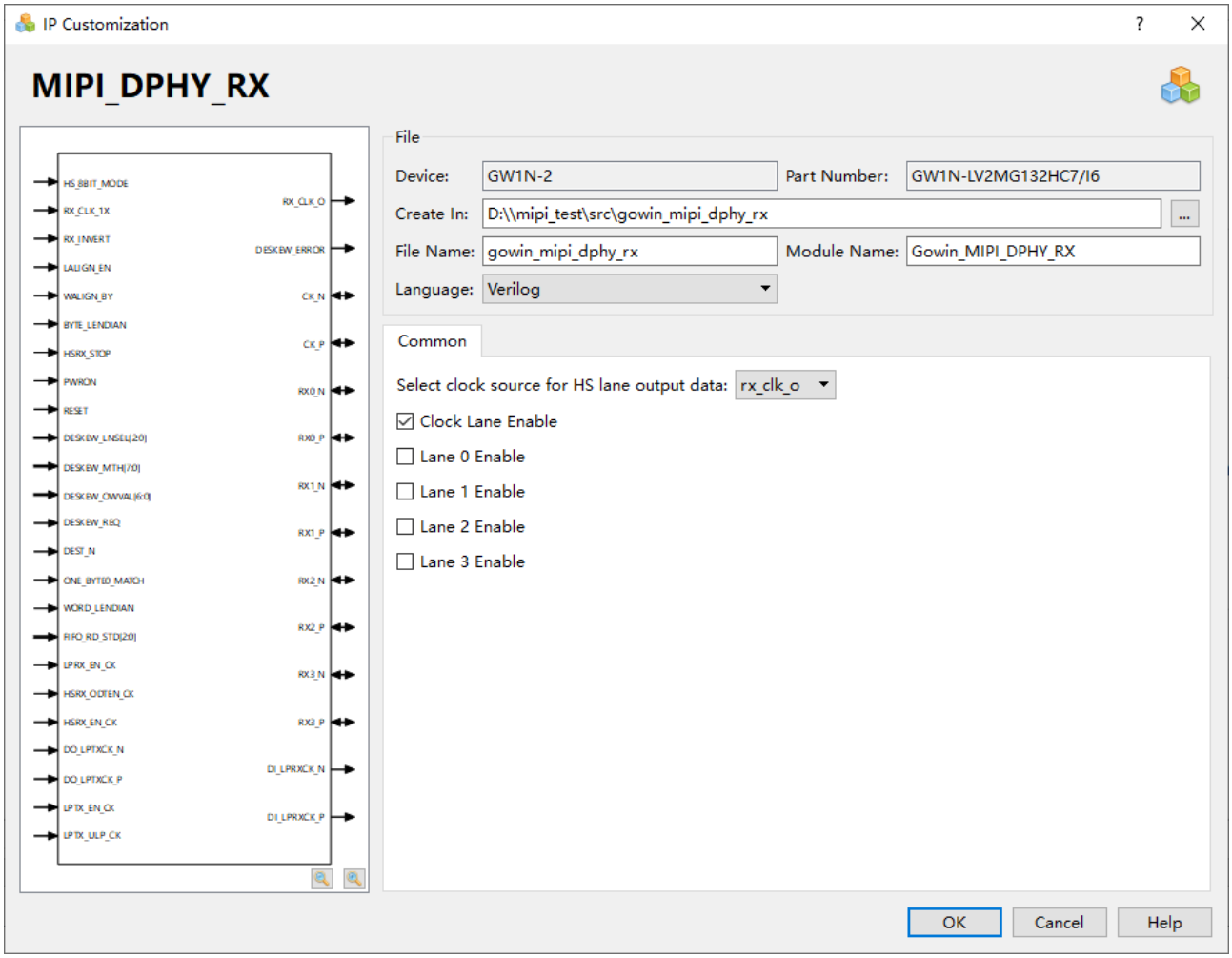
4 MIPI D-PHY RX 配置及调用

在高云云源软件界面菜单栏 Tools 下，可启动 IP Core Generator 工具，完成调用并配置 MIPI D-PHY RX。

4.1 MIPI D-PHY RX 配置

MIPI D-PHY RX 配置界面如图 4-1 所示。

图 4-1 MIPI D-PHY RX 配置页面



1. File 配置框

   File 配置框用于配置产生的 IP 设计文件的相关信息。

   - Device：显示已配置的 Device 信息；
   - Part Number：显示已配置的 Part Number 信息；
   - Language：配置产生的 IP 设计文件的硬件描述语言。选择右侧下拉列表框，选择目标语言，支持 Verilog 和 VHDL；
   - Module Name：配置产生的 IP 设计文件的 module name。在右侧文本框可重新编辑模块名称。Module Name 不能与原语名称相同，若相同，则报出 Error 提示；
   - File Name：配置产生的 IP 设计文件的文件名。在右侧文本框可重新编辑文件名称；
   - Create In：配置产生的 IP 设计文件的目标路径。可在右侧文本框中重新编辑目标路径，也可通过文本框右侧选择按钮选择目标路径。

2. Common

   - Select clock source for HS lane output data：指定 HS 通道输出数据的时钟源，选择输出时钟 rx_clk_o 或输入时钟 rx_clk_1x；
   - Clock Lane Enable：启用或禁用 clock lane；
   - Lane 0 Enable：启用或禁用 lane 0；
   - Lane 1 Enable：启用或禁用 lane 1；
   - Lane 2 Enable：启用或禁用 lane 2；
   - Lane 3 Enable：启用或禁用 lane 3。

3. 端口显示框

   端口显示框图显示当前 IP Core 的配置结果示例框图，如图 4-1 所示。

4. Help 按钮

   单击“Help”，显示 IP Core 的配置信息的页面。Help 页面包括当前 IP Core 的概要介绍，以及 Options 各项配置的简要说明。

4.2 MIPI D-PHY RX 生成文件

MIPI D-PHY RX 窗口配置完成后，产生以配置文件“File Name”命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_mipi_dphy_rx.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 MIPI_DPHY_RX；
- IP 设计使用模板文件“gowin_mipi_dphy_rx_tmp.v”，为用户提供 IP 设计使用模板文件；

- IP 配置文件：“gowin_mipi_dphy_rx.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

附录 A MIPI D-PHY 速率表

表 A-1 MIPI D-PHY 速率

Resolution	Frame Rate (HZ)	Bits Per Pixel (Bits)	Total Data Rate (Mbps)	Lane Number	Per Lane Bit Rate (Mbps)	Recommended Gearing Ratio (1:N)	Per Lane Fabric Clock (MHz)
FHD 1920x1080p (2200x1125)	60	8	1188	2	594.0	8	74.25
		10	1485	2	742.5	8	92.81
		16	2376	2	1188.0	16	74.25
		18	2673	4	668.3	8	83.53
		24	3564	4	891.0	8	111.38
	120	8	2376	2	1188.0	16	74.25
		10	2970	4	742.5	8	92.81
		16	4752	4	1188.0	16	74.25
		18	5346	8	668.3	16	41.77
		24	7128	8	891.0	8	111.38
UHD 3840x2160p (4400x2250)	30	8	2376	4	594.0	8	74.25
		10	2970	4	742.5	8	92.81
		16	4752	4	1188.0	16	74.25
		18	5346	8	668.3	16	41.77
		24	7128	8	891.0	8	111.38
	60	8	4752	4	1188.0	16	74.25
		10	5940	8	742.5	16	46.41
		16	9504	8	1188.0	16	74.25
		18	10692	8	1336.5	16	83.53


GOWIN高云
智慧逻辑 定制未来